

(12) UK Patent Application (19) GB (11) 2 362 483 (13) A

(43) Date of A Publication 21.11.2001

(21) Application No 0011728.3

(22) Date of Filing 16.05.2000

(71) Applicant(s)
David Michael Victor
7 Wimbourne Road, BOURNEMOUTH, BH2 6LX,
United Kingdom

(72) Inventor(s)
David Michael Victor

(74) Agent and/or Address for Service
Lucas & Co
135 Westhall Road, WARLINGHAM, Surrey, CR6 9HJ,
United Kingdom

(51) INT CL⁷
G06F 17/60

(52) UK CL (Edition S)
G4A AUXX

(56) Documents Cited
<http://www.eLabor.com/products/project.htm>,
eLabor.com Enterprise Project (March 2000)
SPR KnowledgePLAN 3 (1998), <http://www.artemis.com/kpnewversion.pdf> (product factsheet)

(58) Field of Search
UK CL (Edition S) G4A AUSB AUXX
INT CL⁷ G06F 17/60
ONLINE: EPODOC, WPI, JAPIO, ELSEVIER, INSPEC,
TDB, INTERNET

(54) Abstract Title
Method and apparatus for calculating the magnitude of a task

(57) A method is provided for calculating an expected time for writing software for performing a task. The method comprises under the control of a program stored in a data processor:
entering into said data processor data that defines the functional content of the task;
entering into said data processor data that defines local factors relating to the carrying out of the task;
calculating by means of said data processor from the functional content data and the local factors data the number of lines of code that the software is expected to contain;
calculating by means of said data processor from local factors data and the expected number of lines of code the expected time to write the software; and
providing an output indicating at least the expected time.

In tests the method has been found to provide good predictions of the actual time required for particular programming tasks. A program for carrying out the method may be stored on a magnetic or optical disc or may be supplied as a signal containing one or more digital files for down-loading via a local network or via the Internet.


The screenshot shows the 'Spectre' software interface. It has a menu bar (File, Edit, Help) and a toolbar. Below the menu bar is a tabbed interface with tabs for 'Developer', 'General Details', 'Data Access', 'Data Manipulation', 'Program Logic', and 'Other Factors'. The 'General Details' tab is active. It contains a section titled 'Please give a rating for the following:' with three sub-sections: 'Sorting & Merging', 'Test Conditions', and 'Performance Criteria'. Each sub-section has radio buttons for 'Not Applicable', 'Simple', 'Average', and 'Complex'. The 'Performance Criteria' section has 'Simple' selected. To the right of the questionnaire is a 'Results' panel. It contains the text 'Estimate for program NELLY to be written in IBM VS COBOL II'. Below this is a table with the following data:

Size and Effort	
Estimated lines:	4422 (about average)
Estimated lines per day:	54 (high but achievable)
Estimated days:	82
Estimated days at installation average lines per day:	111












Below the table is a button labeled 'Explain Results...'. Underneath the table is a section titled 'Other Information' with the following data:

Time saving through code/design adaptation:	26% (about average)
Complexity (range 1 to 81):	29 (about average)
Knowledge rating not known	
Experience rating not known	
Construction point score:	57 (high)

At the bottom of the interface are three buttons: 'Previous', 'Next', and 'Calculate'.

 Speckle

File Edit Help



Developer | General Details | Data Access | Data Manipulation | Program Logic | Other Factors

To start the estimation process, please enter :-

The name of your program:
The language it will be written in:
Your installation's average lines per day for the above language:
Percentage of this program's code that can be adapted from other sources:
Percentage of this program's design that can be adapted from other sources:

ProgName

IBM VS COBOL II

50

35%

30%

Previous

Next

Calculate

Results

Please specify Data Access on the Data Access tab

000000

Spectre
File Edit Help

General Details | Data Access | Data Manipulation | Program Logic | Other Factors

Do you wish to specify developer details?
☒ Yes ☐ No

Grade
☐ Junior
☐ Average
☐ Senior
☐ Expert

Experience
☐ Low
☐ Average
☐ High

Developer details

Current task knowledge
Available
☐ None ☐ Some ☐ Good ☐ Detailed

Required
☐ None ☐ Some ☐ Good ☐ Detailed

Installation standards knowledge
Available
☐ None ☐ Some ☐ Good ☐ Detailed

Required
☐ None ☐ Some ☐ Good ☐ Detailed

Operating system knowledge
Available
☐ None ☐ Some ☐ Good ☐ Detailed

Required
☐ None ☐ Some ☐ Good ☐ Detailed

Installation hardware knowledge
Available
☐ None ☐ Some ☐ Good ☐ Detailed

Required
☐ None ☐ Some ☐ Good ☐ Detailed

Previous

Next





Calculate

Results

Please specify Data Access on the Data Access tab



Spectre

Spectre	
File Edit Help	
   	
Results	
Developer General Details Data Access Data Manipulation Program Logic Other Factors	
Please give a rating for the following:	
Data Retrieval <input type="radio"/> Not Applicable <input checked="" type="radio"/> Simple <input type="radio"/> Average <input type="radio"/> Complex	Data Restructuring <input type="radio"/> Not Applicable <input type="radio"/> Simple <input checked="" type="radio"/> Average <input type="radio"/> Complex
Data Presentation <input checked="" type="radio"/> Not Applicable <input type="radio"/> Simple <input type="radio"/> Average <input type="radio"/> Complex	
Please specify Conditions on the Program Logic tab	
Previous	Next

2003

Spectre
File Edit Help

Developer |
General Details |
Data Access |
Data Manipulation |
Program Logic |
Other Factors

Please give a rating for the following:

Conditions

☐ Not Applicable
☐ Simple
☐ Average
☐ Complex

Calculations

☐ Not Applicable
☐ Simple
☐ Average
☐ Complex

Linkages

☐ Not Applicable
☐ Simple
☐ Average
☐ Complex

Results

Please specify Program Name on the General Details tab

Previous

Next

Calculate



Diagram 1: Initial array: 4, 1, 3, 2, 5

Diagram 2: After first swap: 1, 4, 3, 2, 5

Diagram 3: After second swap: 1, 2, 3, 4, 5

Diagram 4: After third swap: 1, 2, 3, 4, 5

Diagram 5: Final sorted array: 1, 2, 3, 4, 5

 Spectra File Edit Help																		
Developer General Details Data Access Data Manipulation Program Logic Other Factors																		
Please give a rating for the following:																		
Sorting & Merging <input checked="" type="radio"/> Not Applicable <input type="radio"/> Simple <input type="radio"/> Average <input type="radio"/> Complex		Test Conditions <input type="radio"/> Not Applicable <input type="radio"/> Simple <input checked="" type="radio"/> Average <input type="radio"/> Complex																
Performance Criteria <input type="radio"/> Not Applicable <input checked="" type="radio"/> Simple <input type="radio"/> Average <input type="radio"/> Complex																		
<div> <div>Previous</div> <div>Next</div> <div>Calculate</div> </div>																		
Results Estimate for program NELLY to be written in IBM VS COBOL II																		
Size and Effort <table border="1"> <tr> <td>Estimated lines:</td> <td>4422</td> <td>(about average)</td> </tr> <tr> <td>Estimated lines per day:</td> <td>54</td> <td>(high but achievable)</td> </tr> <tr> <td>Estimated days:</td> <td>82</td> <td></td> </tr> <tr> <td>Estimated days at installation average lines per day:</td> <td>111</td> <td></td> </tr> </table> <div> <div>Explain Results...</div> </div>				Estimated lines:	4422	(about average)	Estimated lines per day:	54	(high but achievable)	Estimated days:	82		Estimated days at installation average lines per day:	111				
Estimated lines:	4422	(about average)																
Estimated lines per day:	54	(high but achievable)																
Estimated days:	82																	
Estimated days at installation average lines per day:	111																	
Other Information <table border="1"> <tr> <td>Time saving through code/design adaptation:</td> <td>26%</td> <td>(about average)</td> </tr> <tr> <td>Complexity (range 1 to 81):</td> <td>29</td> <td>(about average)</td> </tr> <tr> <td>Knowledge rating not known</td> <td></td> <td></td> </tr> <tr> <td>Experience rating not known</td> <td></td> <td></td> </tr> <tr> <td>Construction point score:</td> <td>67</td> <td>(high)</td> </tr> </table>				Time saving through code/design adaptation:	26%	(about average)	Complexity (range 1 to 81):	29	(about average)	Knowledge rating not known			Experience rating not known			Construction point score:	67	(high)
Time saving through code/design adaptation:	26%	(about average)																
Complexity (range 1 to 81):	29	(about average)																
Knowledge rating not known																		
Experience rating not known																		
Construction point score:	67	(high)																

0000000000

METHOD AND APPARATUS FOR CALCULATING THE MAGNITUDE OF A TASK

FIELD OF THE INVENTION

5

This invention relates to a method and apparatus for predicting the time it will take to write code for performing a software task, which is useful in the management of program development. It also relates to a magnetic or optical disc or signal in which instructions for carrying out the above method are stored.

10

BACKGROUND OF THE INVENTION

Various proposals have been put forward for managing program development, and an example of relatively recent thinking is provided by US-A-
15 5878262 (Shoumura, assigned to Hitachi Software Engineering Co) and is based on the idea of providing a resource file database and link information. Although such a system will speed up the task of software development, it does not provide an indication of how much work is involved in the writing of a particular new program and how long it is expected to take.

20

Function point analysis is a technique that was first proposed by Allan J Albrecht in the late 1970's and has been explained by Capers Jones, '*Sizing-up Software*', Scientific American, December 1998. It provides a means for
25 estimating the size of a software project or of an application developed or enhanced by a software project based on the user's view of the functional requirements of the application, but without regard to the technology, design tools or language used. Five basic functions are recognized which may be classified as data functions or transactional functions as follows:

30

Data functions -

- Internal logical files (ILF) – i.e. logical groupings of data in a system maintained by an end user.
- External interface files (EIF) – i.e. logical groupings of data maintained by other users or systems and used only for reference purposes

Transactional functions -

- External inputs (EI) – which may be used to add, change or delete information from an ILF.
- External outputs (EO) – in which maintained or referenced data is retrieved and/or manipulated to produce an output.
- External inquiries (EQ) – in which an output is produced by the direct retrieval of stored information.

Additionally function point analysis recognizes two adjustment factors:

15

- Functional complexity – for each function the number of data elements and unique groupings is counted and compared to a complexity matrix for that function so that the function can be rated as of low, medium or high complexity depending on a count of data element types and file types referenced.

20

- Value adjustment factor – the unadjusted function point count is multiplied by the above factor which takes into account the technical and operational characteristics of the system and is calculated in response to inputs relating to

25

- Data communications
- Distributed data processing
- Performance
- Heavy use
- Transaction rate
- On-line data entry

30

- End-user efficiency
- On-line update
- Complexity of processing
- Re-usability
- 5 ○ Ease of installation
- Ease of operation
- Multiplicity of sites
- Facilitating change.

10 The result of the analysis is an adjusted number of function points that measure the complexity of the software independent of technology or system. Function points are a better measure of inherent complexity than lines of code because different computer languages require different lengths of code to specify the same operation. As indicated in the Scientific American article, the expense of

15 producing software can be compared in terms of function points independent of machine, operating system and language. In an article on the Internet by Linda Smith of Predicate Logic, Inc, "*Function point analysis and its uses*", (www.predicate.com/wp_fp.html), the stated benefits of function point analysis are that it:

- 20
- Measures objectively, consistently and can be audited.
 - Normalizes data for comparison between projects, applications and organizations.
 - Provides one size measurement for all types of applications and
 - 25 businesses.
 - Makes size available early in the project life cycle.
 - Is easily understood, applied, used and obtained.
 - Represents what is delivered to the customer.
 - Provides a basis for communication with the customer.

It will be appreciated that the emphasis of the work to date on functional point analysis has been towards abstract measurement and an idealized measure of productivity. However, it is often required to predict how particular people with individual skills and experience and possibly with gaps in their pre-existing knowledge required for the task will perform an actual task constrained e.g. by the requirement to use a particular language and to work with a pre-existing computer system and software. If the time for the task is under-estimated, then it will over-run, whereas if it is over-estimated, highly skilled people may be under-utilized.

10 In the inventor's opinion, managers in the software industry almost invariably treat task estimation casually, largely because they do not have effective techniques at their fingertips, with inevitable consequences. Slippage is insidious - it sneaks up one day at a time. If ten staff all lose just one day in a single week, a fortnights-worth of extra effort has to be found and paid for. But
15 could the problem be one of faulty perception? Suppose the estimates were seriously adrift, that they did not represent what was realistically attainable. Would it then be fair to criticize people for incurring slippage and to put them under pressure to make up the deficit? Of course not, but that is exactly what happens, with inevitable results: morale takes a knock, corners are cut, quality
20 suffers and errors are introduced which will cost additional time and money to rectify further down the line. And meanwhile managers continue to lurch down the highway of rushed development, watching the signposts change from Estimate to Deadline to Ultimatum to Inquest. The situation just described is endemic in the profession of software development even when task specifications are paragons of
25 completeness and clarity and the staff are competent seasoned professionals. Slippage accrues in small increments, unnoticed until people suddenly realize that they are two weeks short of meeting tomorrow's milestone.

SUMMARY OF THE INVENTION

It is an object of the invention to provide a method and apparatus for automated prediction e.g. at task level of the time required to complete particular projects based on actual resources available. Such a method and apparatus address a crucial element of the endemic estimation problems of the IT industry, i.e. unreliable and inconsistent prediction at task level.

The present invention provides calculating apparatus that uses questionnaire-style panels to feed algorithms that generate two essential indicators for a proposed task:

- **size:** taking account of the programming language in which the task will be carried out, the task's functionality and its complexity;
- **development effort** with due regard to the experience and knowledge of the staff assigned to the task.

The above apparatus can provide an output in which **Size** is expressed in lines of code (LOC) and **Development Effort** is expressed as a time period e.g. days.

In one aspect, the invention relates to the use of function point analysis to derive the number of lines of code for carrying out a predetermined task in a particular language, and thence an expected time to write the code.

The invention also provides a method for calculating an expected time for writing software for performing a task, said method comprising under the control of a program stored in a data processor:

entering into said data processor data that defines the functional content of the task;

entering into said data processor data that defines local factors relating to the carrying out of the task;

calculating by means of said data processor from the functional content data and the local factors data the number of lines of code that the software is expected to contain;

calculating by means of said data processor from local factors data and the
5 expected number of lines of code the expected time to write the software; and
providing an output indicating at least the expected time.

The invention also relates to software for carrying out the above method stored or carried by an optical or magnetic disc or as a signal.

10

BRIEF DESCRIPTION OF THE DRAWINGS

An embodiment of the invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

15

Figs 1-6 are screens that appear during the running of the estimation program, Fig 1 being for general data entry, Fig. 2 being for developer details, Fig. 3 being for data access, Fig. 4 being for data manipulation, Fig. 5 being for program logic and Fig. 6 is for other factors and also shows at the right hand of
20 the screen the result that appears when data entry has been completed.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

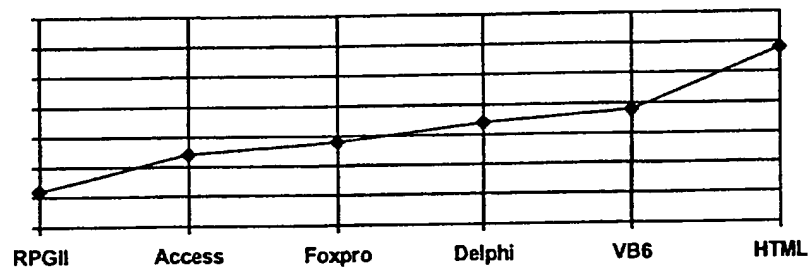
The present task estimator ('SPECTRE) works in two phases:
25

- Function Evaluation
- Development Effort Calculation.

PHASE 1 - Function Evaluation

The starting-point is Function Point Analysis (FPA) which as previously explained is a soundly-based method of quantifying task functionality numerically, derived over many years from industry-wide analyses of software development practice using a variety of programming languages (although debate continues concerning the precise definition of a Function Point, it is nonetheless the *de facto* industry yardstick for software measurement). The rationale of FPA is that a known relationship exists between Function Points and LOC based on the relative power of different programming languages. FPA is traditionally used to measure the size of software in order to compare and benchmark work produced, e.g. the 'Backfire' method for completed projects that quantifies functional content by counting LOC. However, SPECTRE reverses the direction of the Backfire process: it estimates LOC based on specified functional content. The following table shows a few languages for illustration: the higher the position the lower the number of LOC required to generate one point's-worth of function.

RELATIVE POWER OF PROGRAMMING LANGUAGES



Function evaluation recognizes two classes of function:

- **Input/Output**
- **Processing.INPUT/OUTPUT.**

Analysis of the task specification identifies its input/output operations.
SPECTRE currently recognises:

- read from a file
- 5 • write to a file
- delete from a file
- update a file
- select from a table
- insert into a table
- 10 • delete from a table
- update a table
- interface with another program
- generate a report
- generate a display

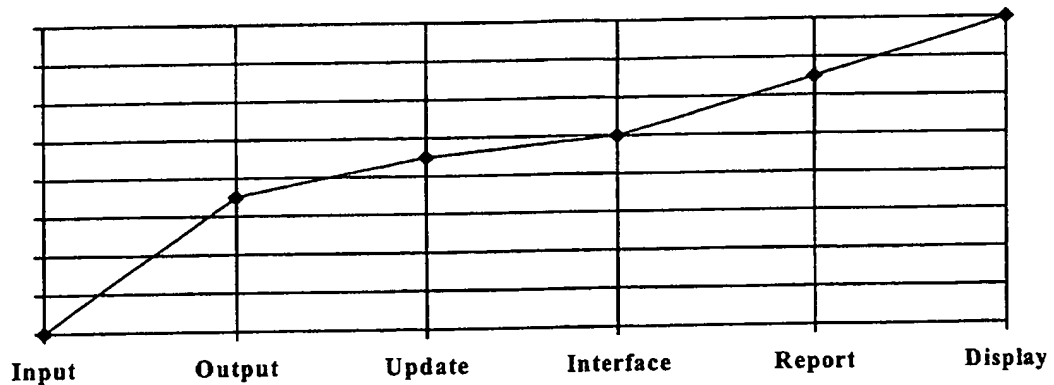
15

Each input/output operation in the task carries its own Function Point cost. These are accumulated, with repeated occurrences of a function shaded down in value (to take account of the probability of shared code for, e.g., error and exception handling) to give a Function Point Total for the task.

20

The following table shows the Function Point cost of input/output operations; the higher the position in the table, the greater the number of Function Points. Please note that the scale shown reflects relative, not actual, Function Points. Note also that the values indicated refer to basic 'function templates' held
25 internally; the preliminary LOC count thus calculated for the task is later adjusted in the light of the task's complexity (see below).

RELATIVE FUNCTION POINT COST OF I/O OPERATIONS



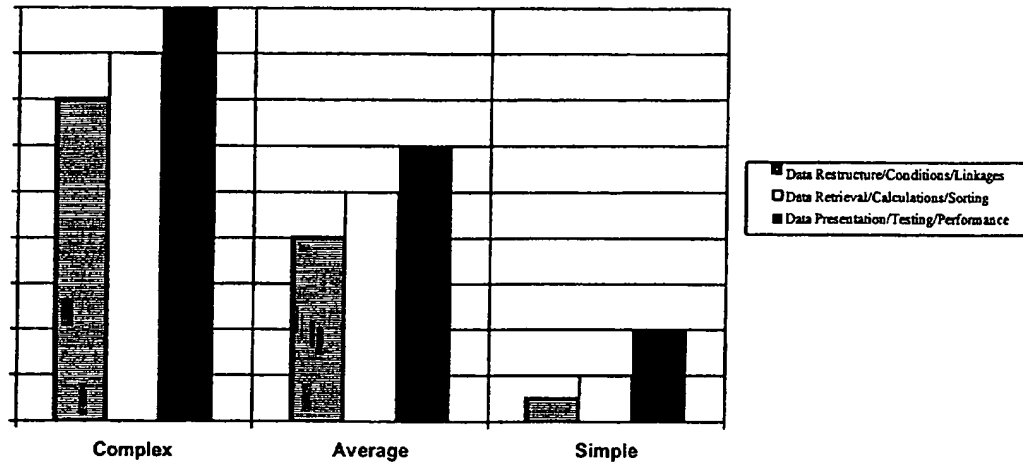
5 PROCESSING

The following components of the processing logic are identified and evaluated:

- data restructure
- 10 • conditions
- linkages with other programs
- data retrieval
- calculations
- sorting
- 15 • data presentation
- testing considerations
- performance considerations

Each applicable feature is given a weighting to represent its degree of
20 complexity, from which a **Complexity Coefficient** is derived for the task. LOC is
calculated as The Function Point Total times the LOC for one point's-worth of
function in the programming language to be used, and the Complexity Coefficient
is applied to this product. The following table shows the relative complexity
weighting of each of the above features; the higher the position in the table, the
25 greater the weighting.

RELATIVE COMPLEXITY OF PROCESSING FEATURES



5 PHASE 2 - Development Effort Calculation

Development Effort Calculation takes into account the grade and experience of the task assignee. Allowance is also made for the possible re-use of existing code and/or program design (all these factors are explained in further detail below). The result is an estimate of the number of days' effort that should be required from receipt of specification to completion of unit-testing (or equivalent acceptance point). Development Effort Calculation requires the following parameters:

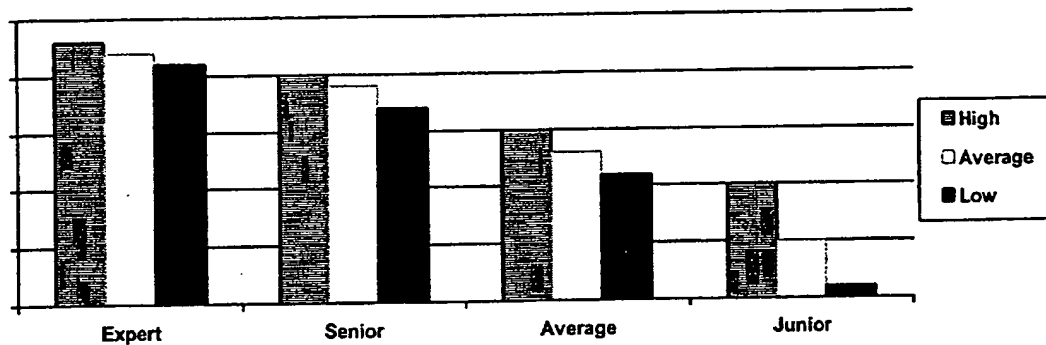
- LOC
- Basic Lines Per Day
- Assignee's Grade
- Assignee's Experience
- Assignee's Knowledge
- Knowledge Required
- Percentage of Design and Code that could be adapted from existing sources

LOC has already been estimated (see above). Basic Lines Per Day is the LOC per day that the installation would expect to achieve for code-and-unit-test as

an average for all development staff. **Grade** is not necessarily a function of job-title; it is an indication of where a neutral observer would place the assignee on a scale ranging from Expert downwards. **Experience** is weighted by reference to a scale ranging from High downwards. The weightings are grade-specific, e.g. a high-experience senior programmer is rated at slightly less than a low-experience expert. Grade and Experience are combined to give an **Experience Coefficient**. The following table illustrates the relative weightings given to Grade and Experience.

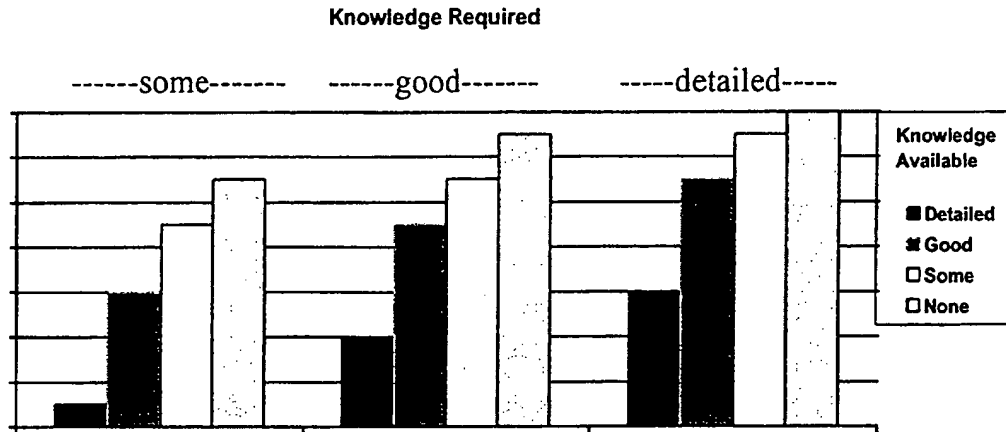
10

RELATIVE ABILITY BY GRADE & EXPERIENCE



Assignee's Knowledge is an assessment of how much the assignee knows, not only of the task in question but also of relevant related subjects. It is broken down into a number of descriptors ranging from **detailed knowledge** of the task and related subjects, to **no knowledge** of the task and little or no general knowledge of related subjects. **Knowledge Required** can be characterized, allowing the Assignee's Knowledge to be weighted as to its relevance. This weighting is applied to the Assignee's Knowledge to give a **Knowledge Coefficient**. The following table illustrates the relative weightings given to the shortfall between Assignee's Knowledge and Knowledge Required; the higher the position in the table, the higher the Knowledge Coefficient.

RELATIVE KNOWLEDGE SHORTFALL



5 Many applications contain tried-and-tested pieces of design and code (for exception-handling, processing transactions against a master-file, and so forth). Percentages of Design and Code that could be adapted from other sources allow the economies of re-use to be taken into account (with an allowance for customisation).

10

 The LOC estimate is divided by the basic lines per day to give the basic number of days. This product is then multiplied by the experience and knowledge coefficients to give the estimated development effort in days for this assignee.

RESULTS

15 SPECTRE's final display is in two sections:

RESULTS - SECTION 1

 This section gives the actual estimate and comprises the following items:

20

- LOC
- Lines per day
- Development effort in days

RESULTS - SECTION 2

This section shows a number of additional items which may be of value as
5 well as interest. These are:

- Time saving through code/design adaptation
- Complexity rating
- Experience rating
- 10 • Knowledge rating
- Function Point score

SPECTRE evaluates all the above in terms of size, range or scope as
appropriate, and appends a comment. Any factor that SPECTRE considers to be
15 significantly outside 'normal' expectation is also flagged with an asterisk.

In use, a new software project is broken down into its component tasks
(each of which will be performed by a single individual) and each task is analyzed
on the basis of its functional content and the relevant local factors, including in
20 particular the grade, experience and knowledge either of the actual person
intended to be assigned to the task or of a person to be hired or recruited. The
results of the analysis in terms of time and lines of code will enable a manager not
only to carry out critical path analysis and other time-sensitive planning tasks but
also to decide e.g. that the level of seniority or experience of the person
25 performing the task should be reconsidered, that a particular task should be
divided, or that other tasks can be combined. In this way and task-by-task a
software development project may be more efficiently planned. Even where a task
definition is not fully established, it can be very beneficial to be able to assess
notional best, average and worst cases. If a task threatens to exceed reasonable
30 limits of size, complexity or effort or looks as if it will require special expertise,
then the earlier these possibilities are flagged the better. The present program is

has been made easy to use to encourage iterative estimating, and as previously indicated, HELP TEXT guidelines have been provided for every step of the estimation process.

5 Figs. 1-6, which are believed to be largely self-explanatory show successive screens in the operation of a practical embodiment of the SCEPTRE software running on a PC under WINDOWS. It will be appreciated that versions of the software can run on other machines, e.g. APPLE machines or machines running UNIX or LINUX. The machine will have the normal computer input and
10 output functions including e.g. a keyboard, a mouse or other pointing device, a CPU, random access memory, a hard disk on which the SPECTRE program will normally be stored and output devices for peripherals e.g. a printer. The machine will also normally have devices for communication with a local area network, a telephone network and the Internet.

15

 In Fig 1 the name of the program is a free-form parameter entered so that it may be displayed 'for the record' on the estimate screen of Fig. 6 which a user may wish to hard-copy for reference or save to a file or database. The program language may be selected from a drop-down menu supplied by the supplier of the
20 SPECTRE program, parameters defining the implications of selecting any particular language being stored and used in the calculation of estimated lines and estimated time. The user is prompted to enter an average number of lines per day that is expected to be written. The program calculates a deviation from this figure, so that the figure initially entered need not be precise. A value in the range 50-70
25 lines per day is typical of many languages. As previously stated, it is very likely that appreciable amounts of existing program code can be adapted for use within the task, and the user is prompted to enter this as a percentage. The a user is also prompted to quantify the percentage of the program that can be adapted from existing sources with particular regard to standard operations, for example data
30 retrieval or exception handling.

With reference to Fig. 2 the user can choose not to supply developer information, in which case the relevant entry fields will not be made available and the program will not be able to take account of experience or knowledge in its estimate. The user will still be able to obtain a development time and size estimate
5 but based only on a deviation from the average lines per day depending on the functionality and complexity of the task. Even if the identity of the task assignee is not known at the time when SPECTRE is run, it is better to supply developer information reflecting a typical profile for a suitable person because this will generate an advance warning that the task is likely to demand a higher than
10 normal level of expertise or will require more effort than expected. In addition to grade and experience fields, there are four fields relating to available and required knowledge of the current task, the operating system, the installation standards and the installation hardware. If there is a shortfall in any area, this does not mean that the task program cannot be written but that time will be needed to obtain the
15 required knowledge, and this time is included in the estimate finally produced.

With reference to the data access screen of Fig. 3, a database access screen includes an area for database access. In the Inputs/Selects field the user is prompted to enter the count of tables/views from which data is read without being
20 updated. In the Outputs/Inserts field he is prompted to enter the count of tables/views into which data is inserted. In the Updates/Deletes field he is required to enter the count of tables/views whose data is updated or deleted. In the File Access area he is required to enter under Inputs/Reads the count of files (datasets) whose records are read without update intent, under Outputs/Writes the count of
25 files (datasets) to which records are added and under Updates/Rewrites/Deletes the count of files (datasets) whose records are updated or deleted. In the Other I/O field under Screens or Displays he is prompted to enter the count of screens or displays that are managed by the task, under Reports the count of reports that the task generates, and under Program Interfaces the count of programs with which
30 the task communicates via parameters.

In the data manipulation screen of Fig. 4, the user is prompted to give a rating for data retrieval, data presentation and data restructuring. Data retrieval relates to data retrieved directly from files and/or databases. If all retrieval is performed for the user e.g. by calls to parameterized middleware functions, then the user should enter Not Applicable. He should select Simple where the data is simple with straightforward relationships and where some editing/conversion is required. He should select Average where there are some interdependencies and a significant amount of editing/conversion is required. He should select Complex where there are multiple sources and/or data types and/or interdependencies and where a significant amount of key-handling and/or editing may be required. In the Data Restructuring field, the user will select Simple where there is little requirement for editing the data, Average where the data has some complexity, a significant amount of editing/conversion is required and there are some different record formats, and Complex where there is a large number of data items and types with a correspondingly large requirement for editing or conversion and numerous record formats. In the Data Presentation field the user will select Simple where there are up to 6 display lines or 25 display/report items and a low number of element types and relationships, Average where there are up to 15 display lines or 60 display/report items, a fair number of element types and relationships and limited user interaction with displays, and Complex where there are more than 15 display lines or more than 60 display/report items, many element types and relationships and extensive user interaction with displays e.g. scrolling/amendment/insertion/deletion.

In the Program Logic screen of Fig. 5 the user is prompted to enter ratings for conditions, calculations and linkages. In the Conditions field, he should enter Simple where there is straight-through control flow and a low proportion of branching logic, average when there is some non-linear logic consisting mainly of IF/THEN/ELSE or CASE constructs and Complex where processing is heavily conditional, logic is affected by timing and/or resource-usage constraints or several logic levels. Calculations should be rated as Simple where they involve

mostly addition and subtraction (e.g. spreadsheet-type with column and/or row totals) or simple multiplication e.g. compound interest calculation, average where IF/THEN/ELSE or equivalents are used or there are straightforward iterative or statistical operations and Complex where the operation is computation-heavy or includes recursion, non-linear calculations, calculus or the like. Linkages are classified as Simple where there are few calls to other (sub)programs and simple parameters, average where there are numerous calls and some parameter conversion/interpretation and complex where there are numerous calls and/or parameter handling requires significant effort.

10

The Other Factors screen of Fig. 6 requires ratings to be entered for sorting and merging, test conditions and performance criteria. In the Sorting and Merging field, a Simple rating should be entered where the data is simple with little requirement for editing or conversion and key data is readily extrapolated, average where key data derivation requires editing or conversion and Complex where there are special exits and/or substantial record selection and/or file merging. Test conditions should be rated Simple where a test setup is straightforward or already in place, logic paths are uncomplicated and good debugging facilities are available, Average if noticeable test setup effort is required and/or there are numerous logic paths of moderate complexity with some special cases and where debugging facilities are fairly useful and Complex where significant test setup effort is required, there are numerous logic paths, some of high complexity, labor-intensive tracking is required and debugging facilities are of limited usefulness. Performance criteria should be rated as simple where there are no requirements beyond normal 'efficient' programming and design, Average where there are some constraints on the use of memory and/or media and/or where response times or speed of execution are important but achievable using normal methods of development and Complex where response times or speed of execution are crucial and a dominant design requirement and/or where there is a strict restart/recovery protocol.

30

ACCURACY

The following table shows how SPECTRE performed with a set of test cases. The 7th and 8th programs in the table were specifically selected to test the ability to handle tasks at both ends of the size spectrum. The 9th and 10th programs were also used to test Development Effort estimation, where SPECTRE predicted 22 days for both against actual times of 23 and 25 days.

<i>LOC - Estimated</i>	<i>LOC - Actual</i>	<i>Deviation %</i>
1287	1338	-3.9
4715	4848	-2.8
5684	5695	-4.7
1560	1551	+0.6
11280	10776	+4.6
5520	5963	-7.5
540	634	-14.8
11424	12590	-10.0
2277	2342	-2.8
4175	4080	+2.3

10

The present program may be supplied on a magnetic or optical disc or as a signal containing one or more digital files for down-loading via a local network or via the Internet. The invention also includes the program recorded or down-loadable as aforesaid.

15

CLAIMS

1. A method for calculating an expected time for writing software for performing a task, said method comprising under the control of a program stored
5 in a data processor:
entering into said data processor data that defines the functional content of the task;
entering into said data processor data that defines local factors relating to the carrying out of the task;
10 calculating by means of said data processor from the functional content data and the local factors data the number of lines of code that the software is expected to contain;
calculating by means of said data processor from local factors data and the expected number of lines of code the expected time to write the software; and
15 providing an output indicating at least the expected time.
2. The method of claim 1, wherein the functional content data entered includes database access data, file access data and input/output data.
- 20 3. The method of claim 1 or 2, wherein the functional content data entered includes data relating to the complexity of data retrieval data restructuring and data presentation.
4. The method of any of claims 1-3, wherein the functional content data
25 entered includes data relating to the complexity of conditions, calculations and linkages.
5. The method of any preceding claim, wherein the functional content data entered includes data relating to the complexity of sorting and merging, test
30 conditions and performance criteria.

6. The method of any preceding claim, wherein the local factors data includes the language in which the program will be written.
7. The method of any preceding claim, wherein the local factors data includes an average number of lines per day to be written.
8. The method of any preceding claim, wherein the local factors data includes a proportion of the program's code that can be adapted from other sources.
9. The method of any preceding claim, wherein the local factors data includes the percentage of the program's design that can be adapted from other sources.
10. The method of any preceding claim, wherein the local factors data includes data representing the seniority and/or experience of the developer.
11. The method of any preceding claim, wherein the local factors data includes data representing the knowledge of the developer.
12. The method of any preceding claim, wherein the output includes the number of lines estimated.
13. Data processing apparatus containing stored instructions for carrying out the method of any preceding claim.
14. An optical or magnetic disc or signal storing instructions for carrying out the method of any of claims 1-12.
15. Use of function point analysis to derive the number of lines of code for carrying out a predetermined task in a particular language, and thence an expected time to write the code.



Application No: GB 0011728.3
Claims searched: 1-15

Examiner: Matthew J. Tosh
Date of search: 10 January 2001

Patents Act 1977 Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:
UK CI (Ed.S): G4A (AUSB, AUXX)
Int CI (Ed.7): G06F 17/60
Other: ONLINE: EPODOC, WPI, JAPIO, ELSEVIER, INSPEC, TDB, INTERNET

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	http://www.eLabor.com/products/project.htm (eLabor.com) Enterprise Project. See product datasheet (March 2000).	
X	SPR KnowledgePLAN v3 (1998). See product fact sheet at http://www.artemispm.com/kpnewversion.pdf .	15 at least

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

This Page is inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☒ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☒ BLURED OR ILLEGIBLE TEXT OR DRAWING
- ☒ SKEWED/SLANTED IMAGES
- ☐ COLORED OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images
problems checked, please do not report the
problems to the IFW Image Problem Mailbox**